马士兵教育Python 全栈文档

咚宝商城项目-首页-接口开发

1. 接口设计

对于接口的设计，我们要根据具体的业务逻辑，设计出适合业务逻辑的接口。
设计接口的思路：

1. 分析要实现的业务逻辑：
   明确在这个业务中涉及到几个相关子业务。
   将每个子业务当做一个接口来设计。
2. 分析接口的功能任务，明确接口的访问方式与返回数据：
   请求方法（如GET、POST、PUT、DELETE等）。
   请求地址。
   请求参数（如路径参数、查询字符串串、表单、JSON等）。
   响应数据（如HTML、JSON等）

2. 商品分类接口

商品分类前端效果图

2.1 商品分类接口设计

1. **请求方式**

   请求方方法：GET
   请求地址：/index/category

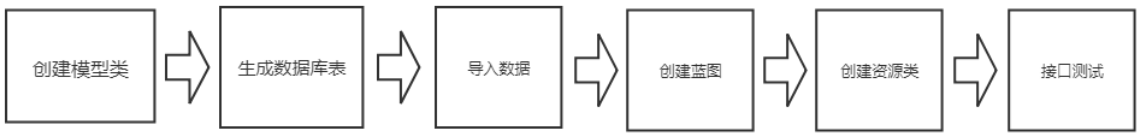
2. **请求参数**

参数名	类型	是否必传	说明
parentid	int	是	上级分类ID, 调用所有分类请传0

3. **响应数据**

   json格式数据

数据接口实现步骤大致如下：



2.2 创建模型类

在comments\models\目录下创建index.py， 然后编写商品分类模型类Category

商品分类模型类代码如下：

```
class Category(db.Model):
    __tablename__ = 't_category'
    __table_args__ = {'comment': '商品分类表'}

    id = db.Column(db.BigInteger, primary_key=True)
    level = db.Column(db.SmallInteger, server_default=text("'1'"), comment='层
级')
    parent_id = db.Column(db.BigInteger, server_default=text("'0'"), comment='父
ID')
    name = db.Column(db.String(255), comment='中文')
    en = db.Column(db.String(255), comment='英语')
    sort = db.Column(db.Integer, comment='排序，暂未使用')
    catid = db.Column(db.Integer, comment='类目id，关联pid使用')
    catid_use = db.Column(db.SmallInteger, server_default=text("'0'"),
comment='是否使用catid查询')
    query_t = db.Column(db.String(255), comment='淘宝查询')
    query_t_use = db.Column(db.SmallInteger, server_default=text("'1'"),
comment='是否使用query')
    weight = db.Column(db.Float, comment='类目单元配重')
    status = db.Column(db.SmallInteger, server_default=text("'1'"), comment='状
态')
    gmt_create = db.Column(db.BigInteger, nullable=False,
server_default=text("'0'"), comment='创建时间')
    gmt_modified = db.Column(db.BigInteger, nullable=False,
server_default=text("'0'"), comment='更新时间')
    create_uid = db.Column(db.String(64), nullable=False,
server_default=text("'0'"), comment='创建人uid')
    create_uname = db.Column(db.String(64), nullable=False,
server_default=text("'0'"), comment='创建人昵称')
    modified_uid = db.Column(db.String(64), nullable=False,
server_default=text("'0'"), comment='更新人uid')
    modified_uname = db.Column(db.String(64), nullable=False,
server_default=text("'0'"), comment='更新人昵称')
```

```
    enabled = db.Column(db.SmallInteger, nullable=False,
server_default=text("'0'"), comment='是否删除:0-未删除;1-删除')
    merchant_id = db.Column(db.String(32), comment='商户ID')
```

2.3 生成数据库表：

在pycharm的终端运行如下命令:

1.将模型添加到迁移文件:

```
python test_migrate.py shopping_db migrate
```

2.迁移文件中的模型映射到数据库中

```
python test_migrate.py shopping_db upgrade
```

2.4 导入商品分类表数据

使用**navicat**运行SQL文件**category.sql**，将商品分类数据导入到t_category表

2.5 创建index蓝图

在Shopping\resources\目录下**创建index包**，用来存放数据接口的资源类

在Shopping\resources\index\目录下的__init__.py文件中**创建名为index的蓝图**

```
from flask import Blueprint

index_bp = Blueprint('index', __name__, url_prefix='/index')  # 创建蓝图
```

在Shopping\目录下的__init__.py文件中通过以下代码**注册蓝图**

```
from Shopping.resources.index import index_bp

app.register_blueprint(index_bp)  #在app中注册蓝图
```

在Shopping\resources\index目录下的__init__.py文件中通过以下代码**创建蓝图中的资源API**

```
from flask_restful import Api
index_api = Api(index_bp)  # 创建蓝图中的资源API
```

在Shopping\resources\index目录下的__init__.py文件中通过以下代码**使用我们自定义json格式**

```
from comment.utils.output import output_json

# 使用我们自定义json格式，替代装饰器的写法
index_api.representation('application/json')(output_json)
```

2.6 创建商品分类的资源类

在Shopping/resources/index/目录中，**创建index_resource.py文件**，我们要创建的资源都放在这个文件中

商品分类的资源类：

```
# 商品分类的资源类
class Shopping_Category(Resource):

    def get(self):
        data = self.getData(0)
        if data:
            for item in data:
                item.update({'list':''})
                data_second = self.getData(int(item['id']))
                item['list'] = data_second
                for item1 in data_second:
                    item1.update({'list': ''})
                    data_third = self.getData(int(item1['id']))
                    item1['list'] = data_third
            return data
        else:
            return {'message':'还没有数据！'}

    #获取分类数据的静态方法
    @staticmethod
    def getData(parent_id):
        res = Category.query.with_entities(Category.id, Category.parent_id,
Category.name).filter(
            Category.parent_id == parent_id).order_by(Category.sort.asc()).all()

        if res:
            data = datalist2dict(res)
            return data
        else:
            return ''
```

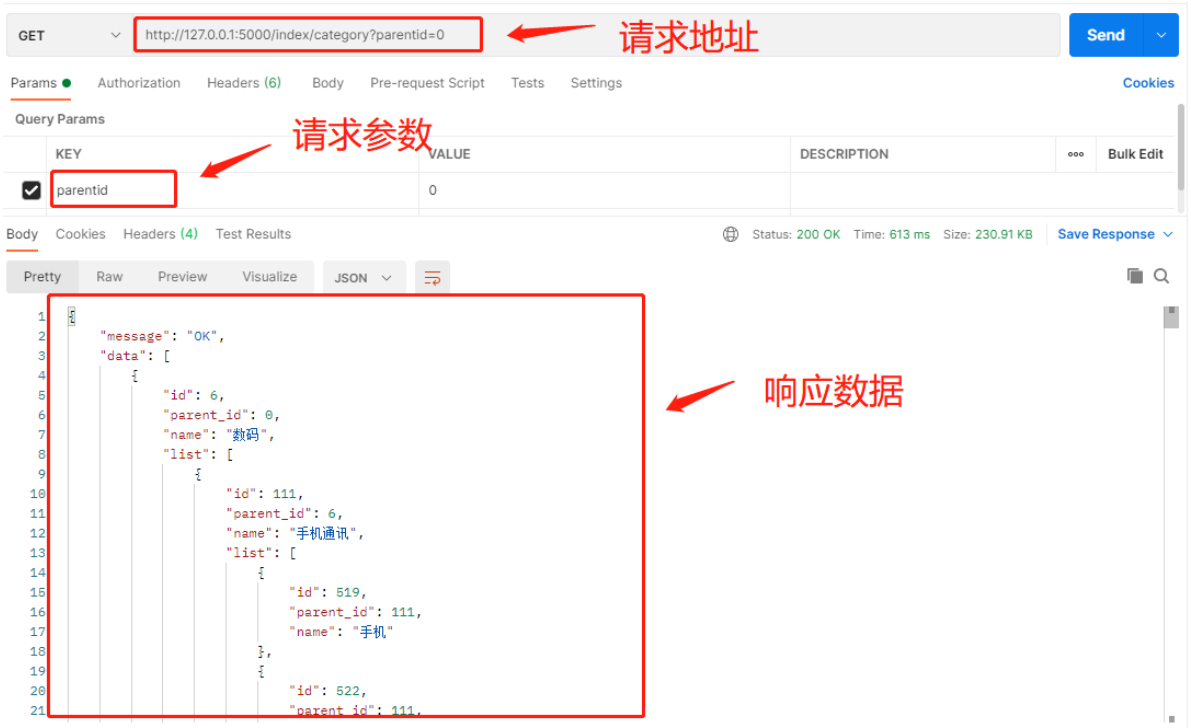
资源类添加到API

```
from Shopping.resources.index.index_resource import *

index_api.add_resource(Shopping_Category, '/category', endpoint='category')
```

2.7 接口测试

使用postmain进行接口测试



3.查询结果转字典函数

在comment\utils目录下新建python文件data2dict.py，此文件主要定义了两个函数，用于查询返回数据转换成字典

函数	功能
datalist2dict(res_obj)	数据库返回数据list 转 dict
data2dict(res_obj)	数据库返回单个数据 转 dict

4. 用Redis对查询数据进行缓存

4.1 为什么使用Redis缓存

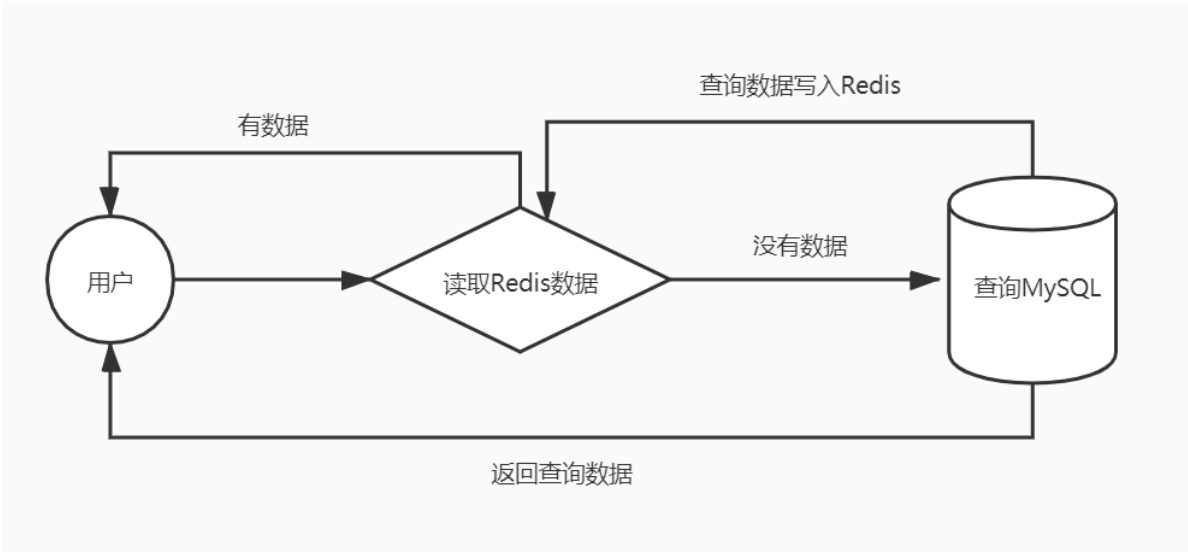
当网站的处理和访问量非常大的时候，我们的数据库的压力就变大了，数据库的连接池，数据库同时处理数据的能力就会受到很大的挑战，一旦数据库承受了其最大承受能力，网站的数据处理效率就会大打折扣。此时就要使用高并发处理、负载均衡和分布式数据库，而这些技术既花费人力，又花费资金。

4.2 Redis缓存原理

Redis其实就是说把表中经常访问的记录放在了Redis中，然后用户查询时先去查询Redis再去查询MySQL，确实实现了读写分离，也就是Redis只做读操作。由于缓存在内存中，所以查询会很快。

4.3 Redis缓存的实现

实现思路如下图所示：

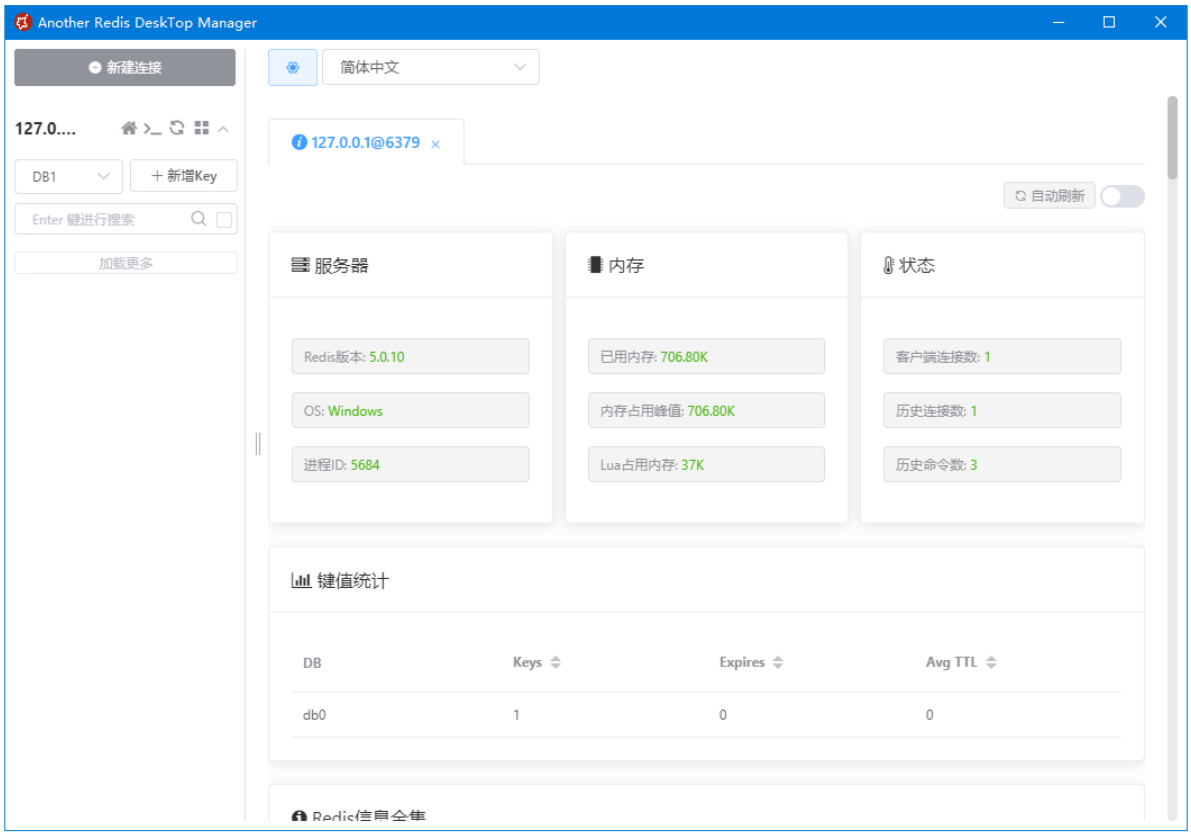


实现代码如下：

```
from comment.utils.shopping_redis import redis_client

#从redis读取数据
data_cache = redis_client.get("index_category")
if data_cache: #如果redis中有数据，则返回redis中的数据
    return json.loads(data_cache)
else:  #如果redis中没有数据，则从mysql查询数据
    pass  #查询mysql数据库，把查询结果写入到redis,并返回数据
```

我们使用**Another Redis Desktop Manager**这款免费开源的Redis客户端工具，来查看和管理Redis数据



5. 首页新品推荐接口

新品推荐前端效果图

5.1 接口设计

1. **请求方式**

   请求方方法：GET
   请求地址：/index/home_new_product

2. **请求参数**

参数名	类型	是否必传	说明
无	无	无	无

3. **响应数据**

   json格式数据

5.2 创建模型类

商品表模型类：

```
#商品表
class Product(db.Model):
    __tablename__ = 't_product'
    __table_args__ = {'comment': '商品表'}

    id = db.Column(db.BigInteger, primary_key=True, comment='商品主键ID')
    product_no = db.Column(db.String(60), comment='商品编号')
    product_name = db.Column(db.String(255), comment='商品名称')
    rel_tenant_id = db.Column(db.BigInteger, comment='商户id')
    rel_default_sku_id = db.Column(db.BigInteger, comment='默认SKU')
    rel_category1_id = db.Column(db.BigInteger, comment='一级类目')
    rel_category2_id = db.Column(db.BigInteger, comment='二级类目')
    rel_category3_id = db.Column(db.BigInteger, comment='三级类目')
    spec_options = db.Column(db.String(60), comment='规格选项id集合')
    price = db.Column(db.Numeric(11, 2), comment='商城价')
    default_pic = db.Column(db.String(512), comment='商品图片')
    album_pics = db.Column(db.Text, comment='商品组图,加上默认主图，最多允许5张图，逗号分割')
    sales_num = db.Column(db.Integer, comment='销量')
    detail_desc = db.Column(db.String(512), comment='商品详情描述')
    publish_status = db.Column(db.SmallInteger, comment='上架状态：0->下架；1->上架')
```

```
    detail_html = db.Column(db.String(512), comment='商品详情web端页面，基本以图为主，富文本HTML样式')
    enabled = db.Column(db.SmallInteger, nullable=False, comment='逻辑删除 0-未删除，1-删除')
    gmt_create = db.Column(db.BigInteger, comment='创建时间')
    gmt_modified = db.Column(db.BigInteger, comment='更新时间')
    modified_uid = db.Column(db.String(50), comment='更新人uid')
    create_uid = db.Column(db.String(50), comment='创建人uid')
    create_uname = db.Column(db.String(255), comment='创建人昵称')
    modified_uname = db.Column(db.String(255), comment='更新人昵称')
```

新品推荐表模型类

```
#新品推荐表
class HomeNewProduct(db.Model):
    __tablename__ = 't_home_new_product'
    __table_args__ = {'comment': '新品推荐表'}

    id = db.Column(db.BigInteger, primary_key=True, comment='主键')
    product_id = db.Column(db.BigInteger, comment='商品ID')
    product_name = db.Column(db.String(128), comment='商品标题')
    recommend_status = db.Column(db.Integer, comment='推荐状态:0-未推荐;1-推荐')
    sort = db.Column(db.Integer, comment='排序')
    gmt_create = db.Column(db.BigInteger, nullable=False,
server_default=text("'0'"), comment='创建时间')
    gmt_modified = db.Column(db.BigInteger, nullable=False,
server_default=text("'0'"), comment='更新时间')
    create_uid = db.Column(db.String(64), nullable=False,
server_default=text("'0'"), comment='创建人uid')
    create_uname = db.Column(db.String(64), nullable=False,
server_default=text("'0'"), comment='创建人昵称')
    modified_uid = db.Column(db.String(64), nullable=False,
server_default=text("'0'"), comment='更新人uid')
    modified_uname = db.Column(db.String(64), nullable=False,
server_default=text("'0'"), comment='更新人昵称')
    enable = db.Column(db.SmallInteger, nullable=False,
server_default=text("'0'"), comment='是否删除:0-未删除;1-删除')
    merchant_id = db.Column(db.String(32), comment='商户ID')
```

5.3 生成数据库表：

通过运行命令生成数据库表（参见2.3）

5.4 导入数据

使用**navicat**运行SQL文件product.sql、home_new_product.sql，将数据分别导入到t_product、t_home_new_product表

5.5 创建新品推荐的资源类

在Shopping/resources/index/index_resource.py文件，编写新品推荐的资源类Shopping_HomeNewProduct

然后将资源类添加到API

```
index_api.add_resource(Shopping_HomeNewProduct, '/home_new_product',
endpoint='home_new_product')
```

5.6 接口测试

使用postman进行接口测试，参见2.7

6.首页人气热搜商品接口

人气热搜前端效果图

6.1 接口设计

1. **请求方式**

   请求方方法：GET
   请求地址：/index/home_recommend_product

2. **请求参数**

参数名	类型	是否必传	说明
无	无	无	无

3. **响应数据**

   json格式数据

6.2 创建模型类

人气热搜商品表模型类：

```
#人气热搜商品表
class HomeRecommendProduct(db.Model):
    __tablename__ = 't_home_recommend_product'
    __table_args__ = {'comment': '人气推荐商品表'}
```

```
    id = db.Column(db.BigInteger, primary_key=True)
    product_id = db.Column(db.BigInteger,comment='商品ID')
    product_name = db.Column(db.String(64),comment='商品名称')
    recommend_status = db.Column(db.Integer,comment='推荐状态')
    sort = db.Column(db.Integer,comment='排序')
    gmt_create = db.Column(db.BigInteger, nullable=False,
server_default=text("'0'"), comment='创建时间')
    gmt_modified = db.Column(db.BigInteger, nullable=False,
server_default=text("'0'"), comment='更新时间')
    create_uid = db.Column(db.String(64), nullable=False,
server_default=text("'0'"), comment='创建人uid')
    create_uname = db.Column(db.String(64), nullable=False,
server_default=text("'0'"), comment='创建人昵称')
    modified_uid = db.Column(db.String(64), nullable=False,
server_default=text("'0'"), comment='更新人uid')
    modified_uname = db.Column(db.String(64), nullable=False,
server_default=text("'0'"), comment='更新人昵称')
    enable = db.Column(db.SmallInteger, nullable=False,
server_default=text("'0'"), comment='是否删除:0-未删除;1-删除')
    merchant_id = db.Column(db.Integer, comment='商户ID')
```

6.3 生成数据库表：

通过运行命令生成数据库表（参见2.3）

6.4 导入数据

使用**navicat**运行SQL文件home_recommend_product.sql，将数据导入到t_home_recommend_product表

6.5 创建人气热搜的资源类

在Shopping/resources/index/index_resource.py文件，编写人气热搜的资源类Shopping_HomeRecommendProduct

然后将资源类添加到API

```
index_api.add_resource(Shopping_HomeRecommendProduct, '/home_recommend_product',
endpoint='home_recommend_product')
```

6.6 接口测试

使用postman进行接口测试，参见2.7

7.首页专题接口

7.1 接口设计

1. **请求方式**

   请求方方法：GET
   请求地址：/index/recommend_subject

2. **请求参数**

参数名	类型	是否必传	说明
无	无	无	无

3. **响应数据**

   json格式数据

7.2 创建模型类

专题表

```
#专题表
class CmsSubject(db.Model):
    __tablename__ = 't_cms_subject'
    __table_args__ = {'comment': '专题表'}

    id = db.Column(db.BigInteger, primary_key=True)
    subject_category_id = db.Column(db.BigInteger, comment='专题分类id')
    title = db.Column(db.String(100), comment='专题标题')
    pic = db.Column(db.String(500), comment='专题主图')
    product_count = db.Column(db.Integer, comment='关联产品数量')
    recommend_status = db.Column(db.Integer, comment='推荐状态')
    create_time = db.Column(db.DateTime)
    collect_count = db.Column(db.Integer)
    read_count = db.Column(db.Integer)
    comment_count = db.Column(db.Integer)
    album_pics = db.Column(db.Text, comment='画册图片用逗号分割')
    description = db.Column(db.String(1000))
    show_status = db.Column(db.Integer, comment='显示状态：0->不显示；1->显示')
    content = db.Column(db.Text)
    forward_count = db.Column(db.Integer, comment='转发数')
    hot_words = db.Column(db.String(500), comment='专题热词,聚类出频率最高的前10个词,
逗号分割')
    gmt_create = db.Column(db.BigInteger, nullable=False,
server_default=text("'0'"), comment='创建时间')
    gmt_modified = db.Column(db.BigInteger, nullable=False,
server_default=text("'0'"), comment='更新时间')
    create_uid = db.Column(db.String(64), nullable=False,
server_default=text("'0'"), comment='创建人uid')
    create_uname = db.Column(db.String(64), nullable=False,
server_default=text("'0'"), comment='创建人昵称')
    modified_uid = db.Column(db.String(64), nullable=False,
server_default=text("'0'"), comment='更新人uid')
    modified_uname = db.Column(db.String(64), nullable=False,
server_default=text("'0'"), comment='更新人昵称')
```

```
    enable = db.Column(db.SmallInteger, nullable=False,
server_default=text("'0'"), comment='是否删除:0-未删除;1-删除')
    merchant_id = db.Column(db.Integer, comment='商户ID')
```

专题分类表

```
#专题分类表
class CmsSubjectCategory(db.Model):
    __tablename__ = 't_cms_subject_category'
    __table_args__ = {'comment': '专题分类表'}

    id = db.Column(db.BigInteger, primary_key=True)
    name = db.Column(db.String(100))
    icon = db.Column(db.String(500), comment='分类图标')
    subject_count = db.Column(db.Integer, comment='专题数量')
    show_status = db.Column(db.Integer)
    sort = db.Column(db.Integer)
    gmt_create = db.Column(db.BigInteger, nullable=False,
server_default=text("'0'"), comment='创建时间')
    gmt_modified = db.Column(db.BigInteger, nullable=False,
server_default=text("'0'"), comment='更新时间')
    create_uid = db.Column(db.String(64), nullable=False,
server_default=text("'0'"), comment='创建人uid')
    create_uname = db.Column(db.String(64), nullable=False,
server_default=text("'0'"), comment='创建人昵称')
    modified_uid = db.Column(db.String(64), nullable=False,
server_default=text("'0'"), comment='更新人uid')
    modified_uname = db.Column(db.String(64), nullable=False,
server_default=text("'0'"), comment='更新人昵称')
    enable = db.Column(db.SmallInteger, nullable=False,
server_default=text("'0'"), comment='是否删除:0-未删除;1-删除')
    merchant_id = db.Column(db.Integer, comment='商户ID')
```

#专题商品关系表

```
class CmsSubjectProductRelation(db.Model):
    __tablename__ = 't_cms_subject_product_relation'
    __table_args__ = {'comment': '专题商品关系表'}

    id = db.Column(db.BigInteger, primary_key=True)
    subject_id = db.Column(db.BigInteger)
    product_id = db.Column(db.BigInteger)
```

7.3 生成数据库表

通过运行命令生成数据库表（参见2.3）

7.4 导入数据

使用**navicat**运行SQL文件cms_subject.sql、cms_subject_category.sql、cms_subject_product_relation.sql，将数据分别导入到t_cms_subject、t_cms_subject_category、t_cms_subject_product_relation表

7.5 创建专题的资源类

在Shopping/resources/index/index_resource.py文件，编写专题的资源类Shopping_RecommendSubject

然后将资源类添加到API

```
index_api.add_resource(Shopping_RecommendSubject, '/recommend_subject',
endpoint='recommend_subject')
```

7.6 接口测试

使用postman进行接口测试，参见2.7